Distributed QoS Evaluation for Real-World Web Services

Zibin Zheng, Yilei Zhang, and Michael R. Lyu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong, China
{zbzheng, ylzhang, lyu}@cse.cuhk.edu.hk

Abstract

Quality-of-Service (QoS) is widely employed for describing non-functional characteristics of Web services. Although QoS of Web services has been investigated in a lot of previous works, there is a lack of real-world Web service QoS datasets for validating new QoS based techniques and models of Web services. To study the performance of real-world Web services as well as provide reusable research datasets for promoting the research of QoS-driven Web services, we conduct several large-scale evaluations on real-world Web services. Firstly, addresses of 21,358 Web services are obtained from the Internet. Then, invocation failure probability performance of 150 Web services is assessed by 100 distributed service users. After that, response time and throughput performance of 5,825 Web services are evaluated by 339 distributed service users. Detailed experimental results are presented in this paper and comprehensive Web service QoS datasets are publicly released for future research.

1. Introduction

Web services have been emerging in recent years and are by now one of the most popular techniques for building versatile distributed systems. The performance of the service-oriented systems is highly relying on the performance of the employed Internet Web services. With the prevalence of Web services on the Internet, investigating quality of Web services is becoming more and more important.

Quality-of-Service (QoS), which is usually employed for describing the non-functional characteristics of Web services, has become an important differentiating point of different Web services [10]. Different Web service QoS properties can be divided into user-independent QoS properties and user-dependent QoS properties. Values of the user-independent QoS properties (e.g., price, popularity, etc.) are usually advertised by service providers and identical for different users. On the other hand, values of the user-dependent QoS properties (e.g., failure probability, response time, throughput, etc.) can vary widely for different users influenced by the unpredictable Internet connections and the heterogeneous user environments.

In the field of service computing [17], a number of QoS driven approaches have been engaged for service selection [8, 15], optimal service composition [2, 5, 16], fault tolerant Web services [7, 18, 19], Web service recommendation [12, 13], and so on. However, there is still a lack of real-world Web service QoS datasets for validating new QoS driven techniques and models. To provide comprehensive studies of the user-independent QoS properties of real-world Web services, evaluations from different geographic locations under various network conditions are usually required. However, it is difficult to conduct large-scale Web service evaluations from distributed locations, since Web service invocations consume resources of the service providers and impose costs for the service users. Moreover, it is difficult to collect Web service QoS data from the distributed service users.

To attack this critical challenge, we conduct several large-scale distributed evaluations on real-world Web services and release reusable Web service QoS datasets for future research. The contributions of this paper are two-fold:

- Firstly, 21,358 Web service addresses are obtained by crawling Web service information from the Internet. Two large-scale distributed evaluations are conducted and first hand experiences on real-world Web service QoS are provided. 1,542,884 Web service invocations are executed by 100 distributed service users on 150 Web services in the first evaluation, while 1,974,675 real-world Web service invocations are executed by 339 distributed service users on 5,825 Web services in the second evaluation. To the best of our knowledge, the scales of our distributed Web service evaluations are the largest in the filed of service computing.
- Secondly, we publicly release our Web service datasets

978-0-7695-4128-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ICWS.2010.10

Figure 1. Locations of Web Services

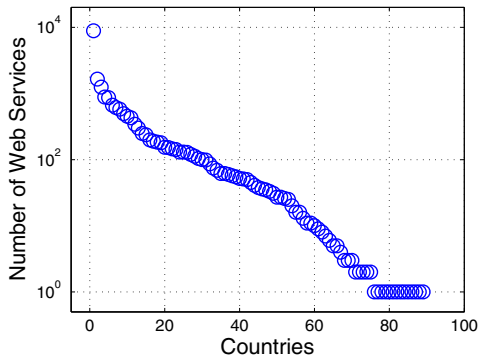


Figure 2. Distribution of Web Services

(e.g., Web service addresses, WSDL files, all the evaluation results, etc.)[1] for future research. The released datasets can be employed by a lot of QoS-aware research topics on Web services.

The rest of this paper is organized as follows: Section 2 introduces the information of Web services. Section 3 investigates failure probability of Web services. Section 4 studies response time and throughput of Web services. Section 5 introduces related work and Section 6 concludes the paper.

2. Information of Web Services

2.1. Crawling Web Service Information

Web services can be discovered from UDDI (Universal Description, Discovery and Integration, which is an XML-based registry enabling companies to publish and discover Web services on the Internet), Web service portals (e.g., *xmethods.net*, *webservicex.net*, *webservicelist.com*, etc.), and Web service searching engines [9] (e.g., *seekda.com*, *esynaps.com*, etc.). By crawling Web service information with these mechanisms at Aug. 2009, we obtain 21,358 addresses of WSDL (Web Service Description Language) files, which provides XML-based descriptions of Web service interfaces. Seekda.com [9] reports that there are totally 28,529 public Web services in the Internet. We believe that the 21,358 Web services in our experiments already cover most of the real-world Web services which are publicly available on the Internet.

[1] http://www.wsdream.net

Table 1. WSDL File Download Failures

Code	Description	# WS	Percent
400	Bad Request	173	3.57%
401	Unauthorized	106	2.19%
403	Forbidden	153	3.16%
404	File Not Found	1468	30.31%
405	Method Not Allowed	1	0.02%
500	Internal Server Error	505	10.43%
502	Bad Gateway	51	1.05%
503	Service Unavailable	22	0.45%
504	Gateway Timeout	788	16.27%
505	HTTP Version Not Support	1	0.02%
N/A	Connection Timed Out	774	15.98%
N/A	Read Timed Out	787	16.25%
N/A	Unknown Host	12	0.25%
N/A	Redirected Too Many Times	3	0.06%
Total		4844	100.00%

By analyzing WSDL files, locations of the Web services can be identified. As shown in Figure 1, these Web services are distributed all over the world, while most Web services are located in North America and Europe. Figure 2 shows the number of Web services provided by different countries. As shown in Figure 2, the Web service numbers of different countries follow the heavy-tailed distribution. Most countries provide a small number of Web services, while a small number of countries providing a large number of Web services. Among all the 89 countries, the top 3 countries provide 55.5% of the 21,358 obtained Web services. These 3 countries are United States (8,867 Web services), United Kingdom (1,657 Web services), and Germany (1,246 Web services). More detailed information of these Web services (e.g., addresses, locations, provider name, etc.) are available in our released datasets.

2.2. Obtaining WSDL Files

By establishing HTTP connections to the 21,358 WSDL addresses obtained in Section 2.1, we successfully download 16,514 (77.32%) WSDL files. The WSDL download failures are summarized in Table 1, where the first column lists the HTTP code indicating different types of failures. The HTTP codes of the last four failure types in Table 1 are non-available (N/A), since we fail to establish HTTP connections and thus unable to obtain the server returned HTTP codes. As shown in Table 1, there are totally 4,844 failures. 48.49% of these failures are timeout failures caused by net-

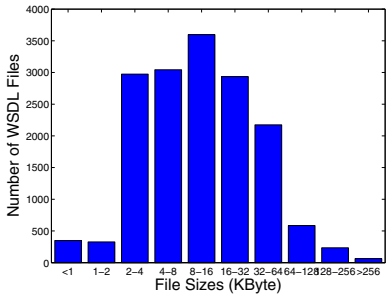


Figure 3. Distribution of WSDL File Sizes

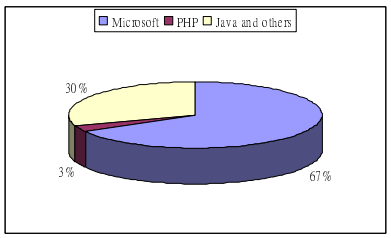


Figure 4. Development Technologies

work connection problems, including 788 (16.27%) *Gateway Timeout*, 774 (15.98%) *Connection Timed out*, and 787 (16.25%) *Read Timed out*. Beside the timeout failures, there is also a lot of *File Not Found* failures (30.31%) and *Internal Server Error* failures (10.43%). The *File Not Found* failures are caused by the removal of WSDL files or update of WSDL addresses, while the *Internal Server Error* failures are caused by the fact that the servers encountered unexpected conditions which prevented them from fulfilling the request. The various types of WSDL file download failures shown in Table 1 indicate that WSDL files on the Internet can become unavailable easily. This highly unavailability of WSDL files are caused by the facts that: (1) the Internet is highly dynamic and unpredictable, (2) the Web service information on the Internet are out-of-date, and (3) many Web services made for experimental purposes.

The WSDL file size distribution can provide an approximate overview of the current status of real-world WSDL files. To achieve this task, we calculate the sizes of the 16,514 downloaded WSDL files and plot the histogram of the WSDL file size distribution in Figure 3. The average size of the obtained WSDL files is 21.981 KBytes. As shown in Figure 3, 90.5% WSDL files are between 2 KBytes to 64 KBytes in size, while there are only 676 WSDL files smaller than 2 KBytes and 883 WSDL files larger than 64 KBytes in size.

Although Web services are black-box to service users without any internal design and implementation details, we can determine their development technologies by analyzing URLs of the WSDL files. For example, WSDL documents generated by Microsoft .NET are usually ended with *".asmx?WSDL"*. We find out that the majority of the collected 16,514 Web services are implemented by Microsoft .NET technology. As shown in Figure 4, 67% of the Web services are implemented by Microsoft .NET technology, 3% are developed by PHP technology, and 30% are implemented by Java and other technologies.

Table 2. Java Code Generation Failures

Failure Type	# WS	Percent
Empty File	249	7.31%
Invalid File Format	1232	36.17%
Error Parsing WSDL	1135	33.32%
Invocation Target Exception	764	22.43%
Null QName	22	0.65%
Databinding Unmatched Type Exception	4	0.12%
Total	3406	100%

2.3. Generating Java Invocation Codes for Web Services

Employing Axis2[2], we successfully generate client-side Web service invocation Java codes for 13,108 (79.38%) Web services among all the 16,514 Web services. Totally 235,262,555 lines of Java codes are produced. There are 3,406 code generation failures, which are summarized in Table 2. As shown in Table 2, among all the 3,406 generation failures, 249 *Empty File* failures are caused by the fact that the obtained WSDL files are empty; 1,232 *Invalid File Format* failures are due to that these WSDL files do not follow standard WSDL format; and 1,135 *Error Parsing* failures are caused by the syntax errors of the WSDL files. There are also 22 *Null QName* failures and 4 *Databinding Unmatched Type* failures. These generation failures indicate that the WSDL files on the Internet are fragile, which may contain empty content, invalid formats, invalid syntaxes, and other various types of errors.

3. Failure Probability

3.1. Dataset Description

To provide objective evaluations on failure probability of the real-world Web services, we randomly select 100 Web services from the 13,108 Web services obtained in Section 2.3 without any personal selection judgments. To conduct distributed evaluations on the selected Web services, we employ 150 computers in 24 countries from PlanetLab [6], which is a distributed test-bed made up of computers all over the world. To make our Web service evalua-

[2]http://ws.apache.org/axis2

Table 3. Statistics of the Dataset 1

Statistics	Values
Num. of Web Service Invocations	1,542,884
Num. of Service Users	150
Num. of Web Services	100
Num. of User Countries	24
Num. of Web Service Countries	22
Range of Failure Probability	0-100%
Mean of Failure Probability	4.05%
Standard Deviation of Failure Probability	17.32%

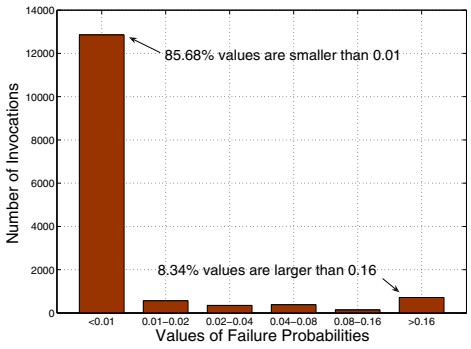


Figure 5. Distribution of Failure Probabilities

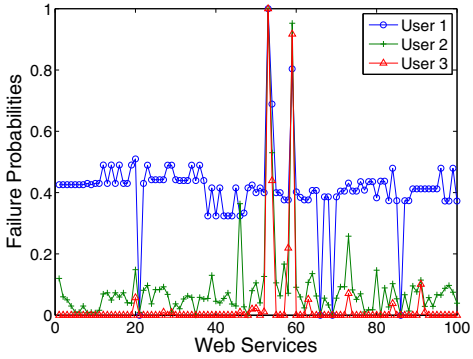
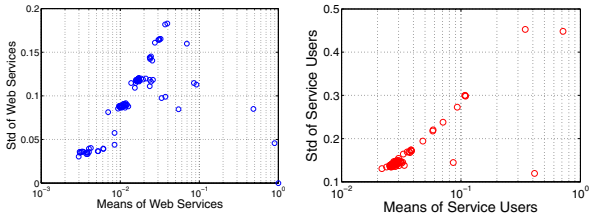


Figure 6. Three Users' Failure Probabilities



(a) (b)

Figure 7. Average Failure Probabilities

tion reproducible, Axis2 is employed for generating client-side Web service invocation codes and test cases automatically. The detailed experimental raw data (e.g., Web service requests, lists of service users and Web services, Web service invocation results, etc.) are provided online[3]. In this experiment, each service user invokes all the 100 selected Web services for about 100 times and records the non-functional performance (i.e., response time, response data size, response HTTP code, failure message, etc.). Totally 1,542,884 Web service invocation results are collected from the service users.

By processing the experimental results, we obtain a 100×150 failure probability matrix, where an entry $f_{a,i}$ in the matrix is the failure probability of Web service i observed by the service user a. In this paper, failure probability $f_{a,i}$ is defined as the probability that an invocation on Web service i by user a will fail. Value of $f_{a,i}$ can be approximately calculated by dividing the number of failed invocations by the total number of invocations conducted by user a on Web service i. As shown in Table 3, the range of failure probability is from 0 to 100%, where 0 means that no invocation fails and 100% indicates that all invocations fail. The mean and standard deviation of all the 15,000 failure probabilities observed by 100 users on 150 Web services are 4.05% and 17.32%, respectively, indicating that the failure probabilities of different Web services observed by different service users exhibit a great variation. Figure 5 shows the value distribution of failure probabilities. As shown in Figure 5, although 85.68% of all the failure probability values are smaller than 1%, a large part (8.34%) of failure probabilities still encounter poor performance with values larger than 16%.

To provide more comprehensive illustration of the Web service failure probabilities observed by different service users, we randomly select three service users (User 1 in US, User 2 in Finland, and User 3 in Germany) from the 150 service users in this experiment and plot their observed failure probabilities of the 100 Web services in Figure 6. As shown in Figure 6, these service users have quite different usage experiences on the same Web services. Failure probabilities of user 1, user 2 and user 3 are around 40%, 10%, and 0% on most of the Web services. The high failure probability of user 1 is caused by the poor client-side network condition. This experimental observation indicates that different users may have quite different usage experiences on the same Web services, influenced by the network connections.

3.2. Overall Failure Probability

To investigate the overall failure probabilities of different Web services, mean of failure probability of Web service i

[3]http://www.wsdream.net

is calculated by:

$$\overline{f_i} = \frac{1}{m}\sum_{a=1}^{m} f_{a,i}, \quad (1)$$

where $f_{a,i}$ is the failure-probability of Web service i observed by the service user a, m is the number of service users ($m = 150$ in this experiment), and $\overline{f_i}$ is the average failure probability of Web service i. Standard deviation of failure probability of Web service i is calculated by:

$$s_i = \sqrt{\frac{1}{m}\sum_{a=1}^{m}(f_{a,i} - \overline{f_i})^2}, \quad (2)$$

where $\overline{f_i}$ is the average failure probability of Web service i and s_i is the standard deviation of failure probability of Web service i.

Similarly, the average failure probability of a service user a can be calculated by:

$$\overline{f_a} = \frac{1}{n}\sum_{i=1}^{n} f_{a,i}, \quad (3)$$

where n is the number of Web services ($n = 100$ in this experiment) and $\overline{f_a}$ is the mean of service user a. Standard deviation of failure probability of service user a can be calculated by:

$$s_a = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(f_{a,i} - \overline{f_a})^2}, \quad (4)$$

where $\overline{f_a}$ is the mean of service user a and s_a is the standard deviation of service user a.

Figures 7(a) and (b) show the mean and standard deviation of the 100 Web services and 150 service users, respectively, where the x axis of the figure is the mean value and the y axis is the standard deviation value. Figure 7(a) shows that: (1) Average failure probabilities of all of the 100 Web services are larger than 0, indicating that 100% invocation success rate is very difficult to achieve in the unpredictable Internet environment, since Web service invocation failures can be caused by client-side errors, network errors, or server-side errors. (2) The standard deviation first becomes larger with the increase of mean and begins to decrease after a certain threshold. This is because the Web services with very large average failure probabilities are usually caused by the server-side errors. The value variation of these Web services to different users is thus not large. For example, there is a Web service with 100% failure probability (caused by the unavailability of that Web service) in Figure 7(a). The standard deviation of this Web service is 0, since all the users obtained the same failure probability, i.e., 100%. (3) Although average failure probabilities of most Web services are small, the standard deviations are quite large, indicating that failure probability values of the same Web service observed by different service users can vary widely.

Figure 7(b) shows that: (1) Average failure probabilities of all of the 150 service users are all larger than 0, although the they are in different locations under various network conditions. This observation indicates that Web service invocation failures are difficult to be avoided on the Internet environment. (2) There is an outlier in Figure 7(b) which has large mean value (0.412) and very small standard deviation value (0.12). This is because most failures (i.e., *UnknownHostException*) of this service user happen to all the other Web services, making the observed failure probabilities on different Web services quite similar. (3) Although average failure probabilities of most service users are small, the standard deviations of most of them are quite large, indicating that failure probability of different Web services observed by the same service user are also quite different.

Table 4. Failures of the Dataset 1

Descriptions	Number
(400)Bad Request	3
(500)Internal Server Error	26
(502)Bad Gateway	33
(503)Service Unavailable	609
java.net.SocketException: Network is unreachable	3
java.net.SocketException: Connection reset	1175
java.net.NoRouteToHostException: No route to host	415
java.net.ConnectException: Connection refused	619
java.net.SocketTimeoutException: Read timed out	4606
java.net.UnknownHostException	5847
java.net.SocketTimeoutException: Connect timed out	44809
Other errors	39
Total	58184

3.3. Failure Types

To investigate different Web service invocation failures, HTTP codes of the Web service responses are employed for the failure detection (i.e., HTTP code 200 indicates invocation success while other codes and exceptions stand for various types of failures). In some special cases, Web service responses with HTTP code 200 may include functional failure information (e.g., invalid parameter, etc.). Such Web service invocations are considered successful, since the target Web services are operating correctly. Since this paper only focuses on non-functional performance evaluation, functional testing of Web services is not considered in this paper. As shown in Table 4, among all the 1,542,884 Web service invocations, there are 58,184 invocation failures. The detailed failures information are summarized in Table 4 and descriptions of different failure types are introduced as

follows:

- *(400)Bad Request*: The Web server was unable to understand the request since the client request did not respect the HTTP protocol completely.
- *(500)Internal Server Error*: The Web server encountered an unexpected condition that prevented it from fulfilling the client request.
- *(502)Bad Gateway*: A gateway or proxy server received an invalid response from an upstream server it accessed to fulfill the request.
- *(503)Service Unavailable*: The Web server was unable to handle the HTTP request due to a temporary overloading or maintenance of the server.
- *Network is unreachable*: A socket operation was attempted to an unreachable network, it didn't get a response and there was no default gateway.
- *Connection reset*: The socket was closed unexpectedly from the server side.
- *NoRouteToHostException*: Socket connection failed caused by intervening firewall or intermediate router errors.
- *Connection refused*: An error occurred while attempting to connect a socket to a remote address and port. Typically, the connection was refused remotely (e.g., no process was listening on the remote address/port).
- *Read timed out*: Timeout occurred on socket read
- *UnknownHostException*: The IP address of a host could not be determined.
- *Connect timed out*: A timeout has occurred on a socket connect.
- *Other failures*: The type of these invocation failures cannot be identified due to lack of failure information.

As shown in Table 4, about 85% of these failures are due to socket connection problems, including 44,809 *connect timed out* and 4,606 *read timed out*. These timed out exceptions are caused by network connection problems during socket connection and socket read. In this experiment, all Web service invocations are configured with a timeout of 20 seconds, which is the default setting of Axis2. By setting a larger timeout value, the number of invocation failures may decrease. The investigations of invocation timeout settings will be conducted in our future work. Besides the timeout exceptions, there are also a lot of other failures caused by network errors, including 33 *bad gateway*, 3 *network is unreachable*, 415 *no route to host*, and 5847 *unknown host*.

Table 5. Statistics of the Dataset 2

Statistics	Values
Num. of Web Service Invocations	1,974,675
Num. of Service Users	339
Num. of Web Services	5,825
Num. of User Countries	30
Num. of Web Service Countries	73
Mean of Response Time	1.43 s
Standard Deviation of Response Time	31.9 s
Mean of Throughput	102.86 kbps
Standard Deviation of Throughput	531.85 kbps

These failures together with the timeout failures account for a large percentage (95.5%) of the Web service invocation failures, indicating that the Web service invocation failures are mainly caused by network errors. Some failures in Table 4 are caused by server-side errors, including 3 *bad request*, 26 *internal server error*, 608 *service unavailable*, 1175 *connection reset*, and 619 *connection refused*. Compared with the failures caused by network errors, the number of failures caused by server-side errors is very small.

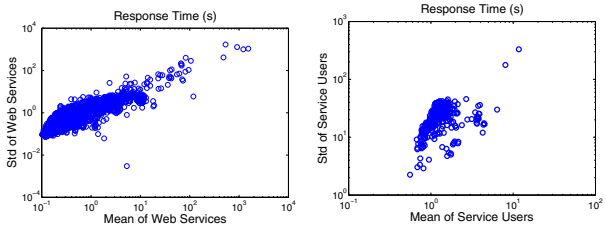
These experimental observations on invocation failures show us that: (1) Web service invocations can fail easily, which can be caused by gateway errors, networking errors, and server errors. (2) In the service-oriented environment, providing reliable Web services is not enough for building reliable service-oriented system, since most invocation failures are caused by network errors. (3) Since the Web service invocation failures are unavoidable in the unpredictable Internet environment, service fault tolerance approaches [11, 19] are becoming important for building reliable service-oriented systems. (4) To tolerate invocation failures caused by network errors, service fault tolerance mechanisms should be developed at the client-side.

4. Response Time and Throughput

4.1. Dataset Description

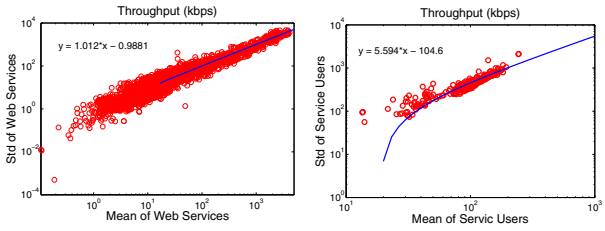
This experiment focuses on investigating the response time and throughput of different Web services and service users. Response time is defined as the time duration between a service user sending a request and receiving the corresponding response, while throughput is defined as the average rate of successful message size (here in bits) delivery over a communication channel per second. This experiment is conducted at Aug. 2009. As shown in Table 5, totally 1,974,675 real-world Web service invocations are executed by 339 service users from 30 countries on 5,825 real-world Web services from 73 countries in this experiment.

By processing the Web service invocation results, we ob-



(a) (b)

Figure 8. Overall Response Time



(a) (b)

Figure 9. Overall Throughput

tain two 339×5825 matrices for response time and throughput, respectively. Each entry in a matrix represents the response time or throughput value observed by a user on a Web service. As shown in Table 5, the mean and standard deviation of response time is 1.43 seconds and 31.9 seconds, respectively, while the mean and standard deviation of throughput is 102.86 kbps and 531.85 kbps, respectively.

4.2. Overall Response-time and Throughput

Figure 8(a) and (b) show the overall response time of Web services and service users, respectively. From Figure 8(a), we observe that: (1) Web services with large average response time tend to have large performance variance to different users, since the standard deviation increases with the mean value in Figure 8(a). (2) Large response time of a Web service can be caused by the long data transferring time or the long request processing time at the server-side. For example, the largest response time (1535 seconds) shown in Figure 8(a) is mainly caused by large size data transferring (12 MBytes data are transferred), while the response time of the outlier (mean = 5.3 seconds, std = 0.0003) in Figure 8(a) is mainly caused by the long request processing time at the server-side. When the response time of a Web service is mainly due to the server-side processing time, different users will receive similar response time, the standard deviation value will thus be small.

Figure 8(b) shows that: (1) Service users with large response time are more likely to observe greater response time variance on different Web services, since the standard deviation increases with the mean value in Figure 8(b). (2) Influenced by the client-side network conditions, different service users observe quite different average response time on the same Web services. Although most service users get good average response time, there is still a small part of service users that receive very large average response time.

Figure 9(a) and Figure 9(b) show the overall throughput value of different Web services and service users, respectively. Figure 9(a) shows that: (1) Similar to the response time, standard deviation of throughput increases with the mean value. (2) Influenced by the poor server-side network conditions, there is a small part of Web services providing a very poor average througput ($<$1 kbps). Figure 9(b) shows that: (1) Influenced by the client-side network conditions, different service users receive quite different average throughput on the target Web services. (2) Service users with large average throughput values are more likely to observe large throughput variance on different Web services, since the standard deviation increases with the mean value.

In Figure 9(a) and Figure 9(b), Two linear functions are fitted to the observed value points. Their equations are also provided. By these equations, performance variance of a Web service (or a service user) can be predicted by their throughput values.

5. Related Work and Discussion

In the field of service computing [17], a lot of QoS driven approaches have been proposed for Web service selection [8, 4, 5, 15], optimal service composition [2, 3, 5, 16], fault tolerant Web services [7, 19], Web service recommendation [21], Web service reliability prediction [20], and so on. However, there is a lack of real-world Web service QoS dataset for verifying these approaches.

In our previous work [18], a real-world Web service evaluation has been conducted by 5 service users on 8 publicly accessible Web services. Since the scale of this experiment is too small, the experimental results are not much useful for future research. Al-Masri et al. [1] release a Web service QoS dataset which is observed by only 1 service user on 2,507 Web services. The fact that different users will observe quite different QoS of the same Web service limits the applicability of this dataset. Our released datasets, on the other hand, include QoS information observed from distributed service users. Moreover, the scales of our datasets are much larger (339×5825 vs 1×2507). Vieira et al. [14] conduct an experimental evaluation of security vulnerabilities in 300 publicly available Web services. Security vulnerabilities usually exist at the server-side and are user-independent (different users observe the same security vulnerabilities on the target Web service). Different from

Vieira's work [14], this paper mainly focuses on investigating performance of user-dependent QoS properties (i.e., failure probabilities, response time, and throughput), which can vary widely among different users.

We believe that without large-scale Web service datasets, characteristics of real-world Web services cannot be fully mined, various service-oriented approaches are thus difficult to be realistic and practical. Our released real-world WSDL file dataset can be employed for research topics such as Web service discovery, WSDL based similarity computation, and so on. Our distributed Web service QoS datasets can be employed for validating various QoS driven approaches of Web services.

6. Conclusion and Future Work

This paper conducts evaluations on user-dependent QoS of Web services from distributed locations. A large number of Web service invocations are executed by service users under heterogenous environments on real-world Web services. Comprehensive experimental results are presented and reusable datasets are released.

In our current Web service evaluations, the invocation timeout is set to be 20 seconds (the default setting of Axis2). More investigations will be conduced to study the relationship between timeout settings and Web service invocation failures. Besides failure probability, response time, and throughput, more user-dependent QoS properties will be investigated in our future work.

Acknowledgement

The authors appreciate the reviewers for their extensive and informative comments for the improvement of this paper. The work described in this paper was fully supported by a grant (Project No. CUHK4154/09E) from the Research Grants Council of Hong Kong, China.

References

[1] E. Al-Masri and Q. H. Mahmoud. Investigating web services on the world wide web. In *Pro. 17th Int'l Conf. World Wide Web (WWW'08)*, pages 795–804, 2008.

[2] M. Alrifai and T. Risse. Combining global optimization with local selection for efficient qos-aware service composition. In *Proc. 18th Int'l Conf. World Wide Web (WWW'09)*, pages 881–890, 2009.

[3] D. Ardagna and B. Pernici. Adaptive service composition in flexible processes. *IEEE Trans. Software Engeering*, 33(6):369–384, 2007.

[4] A. S. Bilgin and M. P. Singh. A daml-based repository for qos-aware semantic web service selection. In *Proc. 2nd Int'l Conf. Web Services (ICWS'04)*, pages 368–375, 2004.

[5] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti. Flow-based service selection for web service composition supporting multiple qos classes. In *Proc. 5th Int'l Conf. Web Services (ICWS'07)*, pages 743–750, 2007.

[6] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, July 2003.

[7] C.-L. Fang, D. Liang, F. Lin, and C.-C. Lin. Fault tolerant web services. *Journal of System Architure*, 53(1):21–38, 2007.

[8] J. E. Haddad, M. Manouvrier, G. Ramirez, and M. Rukoz. Qos-driven selection of web services for transactional composition. In *Proc. 6th Int'l Conf. Web Services (ICWS'08)*, pages 653–660, 2008.

[9] H. Lausen and T. Haselwanter. Finding web services. In *the European Semantic Technology Conf.*, 2007.

[10] D. A. Menasce. Qos issues in web services. *IEEE Internet Computing*, 6(6):72–75, 2002.

[11] N. Salatge and J.-C. Fabre. Fault tolerance connectors for unreliable web services. In *Proc. 37th Int'l Conf. Dependable Systems and Networks (DSN'07)*, pages 51–60, 2007.

[12] L. Shao, J. Zhang, Y. Wei, J. Zhao, B. Xie, and H. Mei. Personalized qos prediction for web services via collaborative filtering. In *Proc. 5th Int'l Conf. Web Services (ICWS'07)*, pages 439–446, 2007.

[13] N. Thio and S. Karunasekera. Automatic measurement of a qos metric for web service recommendation. In *Proc. Australian Software Engineering Conference*, pages 202–211, 2005.

[14] M. Vieira, N. Antunes, and H. Madeira. Using web security scanners to detect vulnerabilities in web services. In *Proc. 39th Int'l Conf. Dependable Systems and Networks (DSN'09)*, pages 566–571, 2009.

[15] T. Yu, Y. Zhang, and K.-J. Lin. Efficient algorithms for web services selection with end-to-end qos constraints. *ACM Trans. the Web*, 1(1):1–26, 2007.

[16] L. Zeng, B. Benatallah, A. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. Qos-aware middleware for web services composition. *IEEE Trans. Software Engeering*, 30(5):311–327, 2004.

[17] L.-J. Zhang, J. Zhang, and H. Cai. Services computing. In *Springer and Tsinghua University Press*, 2007.

[18] Z. Zheng and M. R. Lyu. A distributed replication strategy evaluation and selection framework for fault tolerant web services. In *Proc. 6th Int'l Conf. Web Services (ICWS'08)*, pages 145–152, 2008.

[19] Z. Zheng and M. R. Lyu. A qos-aware fault tolerant middleware for dependable service composition. In *Proc. 39th Int'l Conf. Dependable Systems and Networks (DSN'09)*, pages 239–248, 2009.

[20] Z. Zheng and M. R. Lyu. Collaborative reliability prediction for service-oriented systems. In *Proc. IEEE/ACM 32nd Int'l Conf. Software Engineering (ICSE'10)*, 2010.

[21] Z. Zheng, H. Ma, M. R. Lyu, and I. King. Wsrec: A collaborative filtering based web service recommender system. In *Proc. 7th Int'l Conf. Web Services (ICWS'09)*, pages 437–444, 2009.